MATE : un éditeur de texte basé sur une société d'agents réactifs

Cédric Siléo — Guillaume Hutzler

Laboratoire de Méthodes Informatiques (CNRS - UMR 8042)
Université d'Evry - Val d'Essonne
Tour Evry 2
523 Place des Terrasses de l'Agora
F-91000 Evry

{csileo,hutzler@lami.univ-evry.fr}

RÉSUMÉ. *MATE est un prototype d'Interface Homme/Machine basé sur une société d'agents réactifs et sur un langage de description spatiale de tâches. Implémenté en éditeur de texte, cet outil a pour objectif de montrer que l'on peut construire un logiciel (de type bureautique) en se servant des avantages du paradigme agent et de la puissance des langages de scripts pour rendre l'interface plus personnalisable, extensible et plus intuitive pour des utilisateurs non expert en informatique.*

ABSTRACT. *MATE is a prototype of Computer-Human Interface based on a society of reactive agents and on a language of spatial description of tasks. Implemented as a text editor, this tool aims at showing that we can build a software (of an office automation type) using the advantages of the agent paradigm and the power of script languages in order to make this interface more personalizable, more extendable and more intuitive for non expert users.*

MOTS-CLÉS : *Système Multi-Agent, Interaction Homme/Machine, interfaces, ergonomie, bureautique*

KEYWORDS: *Multi-Agent System, Computer-Human Interaction, interfaces, ergonomics, office automation*

1. Introduction

Ces dernières années ont vu divers logiciels de bureautique comme les traitements de texte, les tableurs, ou encore les gestionnaires de messagerie électronique accroître le nombre et la complexité de leurs fonctionnalités en même temps que leur nombre d'utilisateurs. La cause de ce phénomène est essentiellement due à la vulgarisation de l'informatique qui propose désormais des ordinateurs à la fois puissants et bons marchés. Malgré une forte standardisation des interfaces et les nombreux travaux effectués en Interface Homme/Machine (IHM), ces logiciels, qui font partie de ceux qui sont les plus couramment utilisés, sont toujours construits en priorité autour de leurs fonctionnalités, et non du point de vue de l'utilisateur, sa manière de travailler ou ses habitudes. Ils sont de fait relativement complexes à utiliser et demandent un temps de formation non négligeable. Parmi les problèmes que peuvent poser l'utilisation de ce genre de logiciel, nous avons relevé, d'une manière générale, que :

– les possibilités de personnalisation (options, préférences) sont souvent restreintes,

– l'automatisation des tâches par les macros n'est pas assez robuste,

– l'extension de fonctionnalités requiert des compétences d'expert (programmation directe de script ou de macros),

– l'interface évolue de manière intempestive et incontrôlable (réorganisation de menus ou d'icônes peu pertinente).

Pour tenter de dépasser ces limitations, nous proposons une démarche originale de construction des IHMs fondée d'une part sur l'utilisation d'agents associés aux différents éléments graphiques d'une interface (en s'inspirant de travaux tels que ceux de [REN 01]), et d'autre part sur un langage de description visuelle qui permet de faire évoluer intuitivement l'interface (automatisation de tâches, création de fonctionnalités) en interagissant avec les agents. On espère ainsi redonner à l'utilisateur les moyens d'automatiser et de contrôler simplement les fonctionnalités de son interface, en combinant les avantages du paradigme agent (réactivité, distributivité, simplicité des actions) avec la puissance d'un langage de script et la facilité d'utilisation propre à l'enregistrement de macros. La faisabilité de notre approche a été démontrée grâce au prototype MATE qui implémente un logiciel simple de traitement de texte sous forme d'une société d'agents réactifs. Après avoir décrit rapidement les caractéristiques du système MATE, nous présenterons quelques fonctionnalités redéfinies par un utilisateur lors de la démonstration , pour conclure enfin sur les perspectives d'une telle approche.

2. Brève description de MATE

MATE est un prototype qui permet de formatter du texte en laissant libre choix à l'utilisateur de créer et d'automatiser certaines fonctionnalités propres à un traitement de texte classique, en interagissant avec des agents représentés graphiquement par

des composants de l'interface (boutons, icône, texte...). Un texte étant une structure fortement hiérarchisée (texte, chapitre, paragraphe, mot, lettre) il y a de nombreuses possibilités quant à la représentation graphique d'un agent, sa perception et ses actions. Nous avons arbitrairement fait le choix d'associer un agent à chaque lettre du texte en cours d'édition pour laisser un total contrôle de l'interface à l'utilisateur, notamment pour le placement des lettres et le formattage de paragraphe. Cependant, le nombre d'agents « lettre » (plusieurs dizaines de milliers pour un texte relativement long) oblige nécessairement à restreindre le comportement des agents à des *réactions très simples*. D'autre part, le fait de s'intéresser à un problème de présentation de texte nous a amené à considérer des problèmes d'organisation *spatiale* d'agents. Enfin, pour pouvoir paramétrer facilement l'interface, il est nécessaire de disposer d'un *langage de description* du comportement des agents. Ces contraintes amènent naturellement à s'intéresser aux systèmes qui présentent ce type de caractéristiques, parmi lesquels NetLogo [WIL 99] et les modèles flocking [REY 87]. Nous nous sommes donc inspiré de ces deux systèmes pour construire, dans MATE, un langage de description simple, permettant de décrire les propriétés des agents de l'interface en termes de position, d'orientation, de forme, de taille, de couleur, etc. et à les doter de capacités de perception (voisinage, interaction de l'utilisateur) ainsi que de capacités d'action simples (déplacement, orientation, changement de forme). Ce langage permet non seulement de configurer l'état initial des agents, mais il donne également la possibilité de redéfinir des tâches et des fonctionnalités très intuitivement et progressivement.

L'interface de MATE fonctionne suivant deux modes. Le premier indique à l'agent qu'il doit observer l'utilisateur, ce dernier définissant une nouvelle fonctionnalité en montrant le contexte dans lequel la fonctionnalité doit s'opèrer et les actions qui la caractérisent (primitive `definition`). Le deuxième demande l'exécution par les agents qui ont été sélectionnés de certaines fonctionnalités (primitive `action`). Dans chacun de ces modes, l'interaction entre l'utilisateur et les agents s'effectue par des manipulations avec la souris, manipulations auxquelles est associée une sémantique claire, connue des agents. Lorsque la définition d'une nouvelle fonctionnalité est engagée, la manipulation est constituée de deux phases séquentielles : une pour la description d'un contexte, l'autre pour la définition d'une action. Le contexte désigne l'ensemble des conditions devant être vérifiées pour permettre l'exécution d'une action. Autrement dit, il s'agit d'identifier l'ensemble des perceptions qu'un agent doit avoir pour pourvoir déclencher une action. Dans le mode contexte, deux clics successifs sur deux agents différents définissent ainsi les contraintes de distance et d'angle que doivent observer les agents. Un clic sur l'environnement signifie la perception de l'absence d'agent (perception *vide*). Dans le mode action, il s'agit de faire comprendre à un agent quelle sorte d'action on veut lui faire exécuter, en combinant celles qu'il connaît au départ. Les interactions liées au mouvement et au redimensionnement traduisent les actions que les agents doivent effectuer s'ils correspondent au contexte défini pour la fonctionnalité en cours de définition. Chaque fois qu'une nouvelle fonctionnalité est définie, un nouvel agent-bouton est créé pour permettre l'activation ultérieure de celle-ci (tous les agents la mémorisent globalement). La combinaison de différentes fonctionnalités se fait en activant les boutons définis au fur et à mesure de l'inter-

action. Chaque agent sélectionné par l'utilisateur, lorsqu'il active une fonctionnalité, reçoit l'ordre d'appliquer les actions qu'il a mémorisées précédemment, en fonction des règles de perception qui lui ont été indiquées. Il est ainsi possible de définir des contextes complexes par composition de contextes simples ou encore d'enchaîner des actions. Enfin, pour visualiser la robustesse des nouvelles fonctionnalités par rapport au contexte défini par l'utilisateur, il est possible de déplacer, supprimer ou ajouter des agents au cours du fonctionnement. Grâce à ces moyens d'interaction, il est possible de définir, dans cette application, de nouvelles fonctionnalités plus élaborées grâce à des règles de réactivité chez un ensemble d'agents.

3. Démonstration du fonctionnement de MATE

Le système MATE a permis de redéfinir très simplement de nouvelles fonctionnalités avancées, parmi elles, l'insertion d'une lettrine, l'alignement de texte et la définition d'un titre.

La démonstration consistera, dans un premier temps, à montrer, à l'aide d'exemples et de manipulations, les différents problèmes que posent les logiciels de bureautique actuels (par exemple un traitement de texte) en reprenant les points qui ont été énoncés en introduction. La deuxième partie de la démonstration visera à présenter les différentes étapes permettant de redéfinir une fonctionnalité qui aurait pu poser problème dans la première partie. Elle se composera de la définition incrémentale d'une fonctionnalité en temps réel par description des contextes, puis des actions que l'agent doit effectuer pour réaliser une tâche. La mise en oeuvre de cette partie passera par la définition « personnelle » que l'on attribue à cette tâche. Par exemple, l'insertion d'une lettrine peut se décomposer intuitivement en deux parties : d'abord exhiber le premier caractère d'un paragraphe et augmenter sa taille de police. Ensuite définir le premier caractère par l'intersection de la première ligne et de la première colonne d'un paragraphe. Chaque interprétation des interactions de l'utilisateur par l'agent sera visualisée sous la forme d'un schéma XML représentant les états internes successifs de l'agent. La dernière étape de la démonstration illustrera l'exécution des tâches ainsi définies dans différents contextes d'utilisation, en insistant sur quelques aspects de robustesse gagnés avec cette approche.

4. Conclusion et perspectives

Nous avons proposé, par ce travail, une démarche de construction des IHMs fondée d'une part sur l'utilisation d'agents associés aux différents éléments graphiques d'une interface, et d'autre part sur un langage de description visuel par lequel l'utilisateur peut personnaliser le comportement des agents grâce à une manipulation directe et intuitive des agents. Le système MATE a montré la faisabilité de notre approche en implémentant un logiciel simple de traitement de texte sous forme d'une société d'agents réactifs. L'intérêt d'une telle approche est double : d'une part, elle propose une autre façon de concevoir les interfaces en distribuant les fonctionnalités de celles-ci par le

paradigme agent et elle propose d'autre part une démarche originale de description des comportements d'agents réactifs, sur la base d'un langage de description fondé sur des notions spatiales.

MATE ne constitue que les prémisses d'un projet plus ambitieux qui concerne la construction d'IHM à l'aide d'agents capables de gérer n'importe quelle interface. La robustesse de celle-ci passera nécessairement par une abstraction beaucoup plus forte et approfondie des interactions de l'utilisateur et par un apprentissage par l'exemple des agents (l'utilisation des travaux de [LIE 01] est une piste sur laquelle nous travaillons déjà). Ce travail ouvre la voie à la conception d'IHMs entièrement basées sur le concept agent, interfaces potentiellement beaucoup plus adaptatives et personnalisables. Dans le contexte de l'informatique diffuse qui voit la multiplication des objets "calculants" et "communicants" dans notre environnement quotidien, ce travail trace également des pistes pour trouver de nouveaux paradigmes d'interaction adaptés à ces nouveaux contextes d'interaction qui nous font face. En retour, ce travail de réflexion sur l'interaction pourra également profiter au domaine multi-agent de par les perspectives de conception participative qu'il ouvre.

Remerciements

Ce travail est issu d'un projet de recherche conventionné (CIFRE) entre le Laboratoire de Méthodes Informatique (LaMI), la Société Netizis et le Conseil Général du département de la Haute-Saône. Les auteurs tiennent à les remercier de leur soutien.

5. Bibliographie

[LIE 01] LIEBERMAN H., *Your Wish is my Command : Programming by Example*, Morgan Kaufmann Publishers, San Francisco, 2001.

[REN 01] RENAULT V., DROGOUL A., « LEA : Learning E-mail Agents », *JFIADSMA-01*, Editions Hermès, Paris, 2001, p. 343-345.

[REY 87] REYNOLDS C., « Flocks, Herds, and Schools : A Distributed Behavioral Model », *Computer Graphics*, ACM SIGGRAPH '87 Conference Proceedings, Anaheim, California, juillet1987, p. 25-34.

[WIL 99] WILENSKY U., « U. 1999. NetLogo. http ://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling », , 1999.